L4: Entry 1 of 1 File: USPT Sep 22, 1992

DOCUMENT-IDENTIFIER: US 5150469 A
TITLE: System and method for processor pipeline control by selective signal deassertion

Detailed Description Paragraph Right (14):
The CPU 18 has a four stage pipeline. The stages are instruction fetch (IF) 50, compute (ALU) 52, memory access (MEM) 54 and result write back (WB) 56. The CPU 18 has four major data path blocks: adder/subtractor 57, shifter/extractor 59, logical unit 61 (capable of executing 16 Boolean functions) and register file 63. The register file 63 has three ports: two are read for A and B ALU source operands and the ALU results are written to the W port. There are three instruction registers (IR) 67-71. IR[ALU] 67, IR[MEM] 69 and IR[WB] 71 hold instructions supplied by the instruction buffer 65 in the ALU, MEM and WB pipestages, respectively. The IR 67-71 consist of an opcode and one register specifier. Source operand and bypass multiplexers 73 and 75 are connected between the register file 63 and the adder/subtractor 57, the shifter/extracter 59 and the logical unit 61. PC selection multiplexer 77 is connected to supply results of the ALU operations and instruction addresses that have been incremented by incrementer 81 to an instruction address register (PC) 79. A description of the actions taken in each pipestage 50-56 is given in Table IV below. Each pipestage is broken into four clock phases p1-p4. One and only one of the clock phases is high at all times, subject to small amounts of skew, which may result in zero or two clock phases being high for a few nanoseconds.

Detailed Description Paragraph Table (11):
TABLE X

Stage	Phase	Operation
ALU	1	Read CPU opcode bus
	2 3 4	MEM 1 Coproc ALU instr driven from CPU
	2 3	Set up register file predecoders
	4	Read and latch source registers Send op code to FU's Read register reservation table Compute if this was a valid FPU ALU instr Compute if bypass required - tell FU's If interrupt, nop instr leaving MEM WB
	1	Drive sources to FU's (EX1) Provisionally issue the instruction FU's latch sources or result if source bypass FU's start computing, first FU pipestage
	2	If instr was valid and ready when issued then begin reserve destination reg; increment appropriate register specifiers; decrement vector length
	3	If ((CPU is sending another ALU instr) and (vector length remaining <>-1)) or register interlock then deassert LoadWB
	4	EX2
	1	Continue computing, second FU pipestage
	2 . 3 . 4 .	EX3
	1	Continue computing, third FU pipe- stage
	2 . 3 . 4 .	PA
	1	FU drives result Clear reservation bit
	2	Write result if IR[PA]<>nop
	3	Assert interrupt if IR[PA]<>nop and overflow

Current US Original Classification (1):
712/244

L5: Entry 1 of 1 File: USPT Sep 22, 1992

DOCUMENT-IDENTIFIER: US 5150469 A
TITLE: System and method for processor pipeline control by selective signal deassertion

Detailed Description Paragraph Right (14):
The CPU 18 has a four stage pipeline. The stages are instruction fetch (IF) 50, compute (ALU) 52, memory access (MEM) 54 and result write back (WB) 56. The CPU 18 has four major data path blocks: adder/subtractor 57, shifter/extractor 59, logical unit 61 (capable of executing 16 Boolean functions) and register file 63. The register file 63 has three ports: two are read for A and B ALU source operands and the ALU results are written to the W port. There are three instruction registers (IR) 67-71. IR[ALU] 67, IR[MEM] 69 and IR[WB] 71 hold instructions supplied by the instruction buffer 65 in the ALU, MEM and WB pipestages, respectively. The IR 67-71 consist of an opcode and one register specifier. Source operand and bypass multiplexers 73 and 75 are connected between the register file 63 and the adder/subtractor 57, the shifter/extracter 59 and the logical unit 61. PC selection multiplexer 77 is connected to supply results of the ALU operations and instruction addresses that have been incremented by incrementer 81 to an instruction address register (PC) 79. A description of the actions taken in each pipestage 50-56 is given in Table IV below. Each pipestage is broken into four clock phases p1-p4. One and only one of the clock phases is high at all times, subject to small amounts of skew, which may result in zero or two clock phases being high for a few nanoseconds.

Detailed Description Paragraph Right (32):
The CPU places proper parity on the driven word of the data bus in CPU stores and CPU.fwdarw.Cop transfers. In order to drive the parity bits at the same time as the data bits, the parity bits must be available when the data is read out of the register file (i.e., the parity bits must be stored in the register file too). Data is written into the register file from two sources: the local data bus, or a functional unit within the CPU (e.g., ALU, barrel shifter, and compare unit). The local data bus has parity; to provide proper parity in the case of results of CPU functional unit results written into the register file, parity is computed on the R bus during the ALU, shift or extract instruction's normally idle MEM stage. On each write of the register file the parity of the written word is checked. This checks incoming data in the case of a CPU load-class instruction, and also checks the parity generator in the case of a CPU ALU, shift, or extract instruction. Since parity is stored in the register file, parity errors in the register file as well as the external data busses and data cache can also be detected. However, since the CPU does not check the parity of operands read out of the register file, a parity error in the register file is only detected if the erroneous register is written back to main memory or is written into the data cache and loaded back into the register file.

Detailed Description Paragraph Right (40):
FIG. 4 gives an overview of the microarchitechture of the FPU 20 and its pipeline. The FPU extends the four stage pipeline to a total of seven stages 58-68. The additional pipestages are EX2 64 (execute stage two), EX3 66, and PA 68 (write back ALU result, called put away to prevent confusion with WB). Execute stage one is WB 62. The FPU 20 has four major data path blocks: adder 70, multiplier 72, reciprocal approximation unit 74 and register file 76. The register file 76 has four ports: two are read for A and B ALU source operands, the ALU results are written to the R port, and loads/stores/transfers read or write the memory (M) port. In addition, a time-of-day counter 78, an interval timer 80 and an FPU PSW 82 are above the register file 76. There are eight instruction registers (IR) 84-98. IR[ALU] 84 and

IR[MEM]86 hold instructions in the ALU and MEM pipestages 58 and 60, respectively. IR[WB[LST]] 88 holds load/store/transfer instructions in the WB pipestage 62. Similarly, IR[MEM[ALU]] 90 holds ALU instructions in the MEM pipestage 60, and IR[WB[ALU]] 92 holds ALU instructions in the WB pipestage 62. IR[EX2] 94, IR[EX3] 96 and IR[PA] hold ALU instructions in the EX2, EX3 and PA pipestages 64, 66 and 68, respectively. All instruction registers except IR[MEM[ALU]] 90 and IR[WB[ALU]] 92 consists of an opcode and one register specifier; IR[MEM[ALU]] 90 and IR[WB-[ALU]] 92 contain an entire 32 bit FPU ALU instruction. Source operand and bypass multiplexers 100 and 102 are connected between the register file 76 and the adder 70, multiplier 72 and reciprocal approximation unit 74.

Detailed Description Paragraph Table (11):

TABLE X

Stage	Phase	Operation
ALU	1	Read CPU opcode bus
	2 3 4	MEM 1 Coproc ALU instr driven from CPU
	2 3	Set up register file predecoders
	4	Read and latch source registers Send op code to FU's Read register reservation table Compute if this was a valid FPU ALU instr Compute if bypass required - tell FU's If interrupt, nop instr leaving MEM WB
	1	Drive sources to FU's (EX1) Provisionally issue the instruction FU's latch sources or result if source bypass FU's start computing, first FU pipestage
	2	If instr was valid and ready when issued then begin reserve destination reg; increment appropriate register specifiers; decrement vector length end
	3	If ((CPU is sending another ALU instr) and (vector length remaining <>-1)) or register interlock then deassert LoadWB
	4 EX2	1 Continue computing, second FU pipestage 2 . 3 . 4 .
EX3	1	Continue computing, third FU pipe- stage 2 . 3 . 4 .
PA	1	FU drives result Clear reservation bit
	2	Write result if IR[PA]<>nop
	3	Assert interrupt if IR[PA]<>nop and overflow

Current US Original Classification (1):
712/244

WEST

Freeform Search

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L1 and ((register adj specifier$) same (register file) same (increment$))

Display: 10 **Documents in Display Format:** KWIC **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search | Clear | Help | Logout | Interrupt

Main Menu | Show S Numbers | Edit S Numbers | Preferences | Cases

Search History

DATE: Thursday, April 25, 2002 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L4	L1 and ((register adj specifier$) same (register file) same (increment$))	1	L4
L3	L1 and ((register adj pointer) same (register file) same (increment$))	3	L3
L2	L1 and ((register adj pointer) same (register adj file))	74	L2
L1	((712/$)!.CCLS.)	7305	L1

END OF SEARCH HISTORY

DATE: Wednesday, April 24, 2002 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L3	L1 and ((register adj file) with counter with operand$ with increment$)	10	L3
L2	L1 and ((register adj file) with counter$)	266	L2
L1	((712/$)!.CCLS.)	7305	L1

END OF SEARCH HISTORY

DATE: Wednesday, April 24, 2002 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L11	L9 and counter$	3	L11
L10	L9 and ((operand or data) adj counter$)	0	L10
L9	L8 and ((VLIW) and (register adj file))	7	L9
L8	L7 and ((floating adj point$) and (double adj precision))	59	L8
L7	l1 and (plurality with functional with units)	490	L7
L6	L3 and (floating adj point$)	0	L6
L5	L3 and (functional adj units)	0	L5
L4	L3 and (plurality with functional with units)	0	L4
L3	(5313600 or 5010483 or 4928238 or 4885678 or 4086626).pn.	5	L3
L2	L1 and (((operand or data) adj counter) with increment$)	13	L2
L1	((712/$)!.CCLS.)	7305	L1

END OF SEARCH HISTORY

DATE: Wednesday, April 24, 2002 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L2	L1 and (((operand or data) adj counter) with increment$)	13	L2
L1	((712/$)!.CCLS.)	7305	L1

END OF SEARCH HISTORY